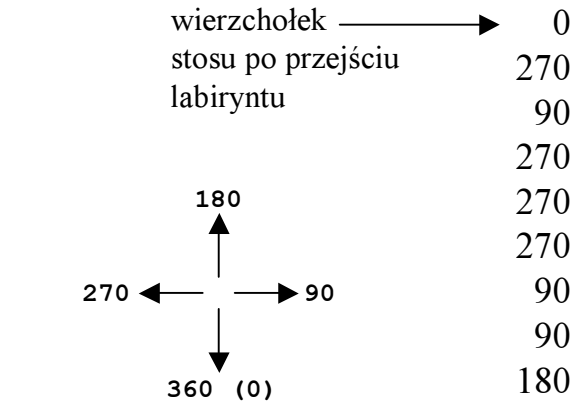
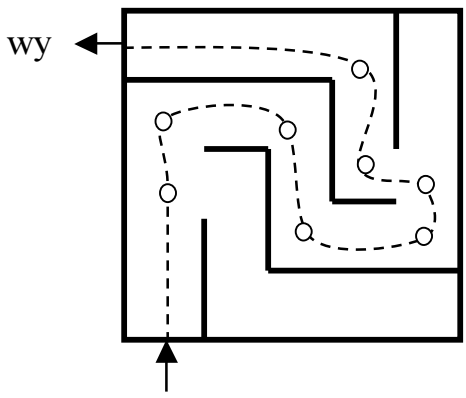
Algorytm przejścia labiryntu

Dany jest robot-pojazd, który ma być używany do znajdowania trasy przejścia przez labirynt (od wejścia do wyjścia).
Po pokonaniu labiryntu, robot ma wydrukować trasę przejazdu/powrotu.

Sterowanie robotem jest możliwe za pomocą:
- procedury **`go()`**, która powoduje przesuwa robota naprzód do napotkania następnego skrzyżowania lub ściany;
- funkcji **`front()`**, która pozwala ocenić co znajduje się przed robotem (ściana czy wyjście);
-procedury **`turn(s)`**, która obraca robota o x-stopni w prawo.

Do zapamiętywania pokonywanej trasy wykorzystywana jest struktura stosu, na której odkładane są kolejne wartości kierunku dla kolejnych skrzyżowań np..

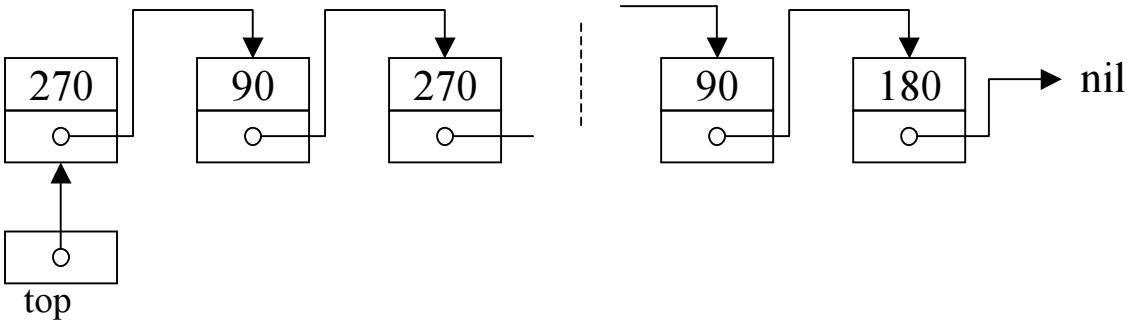




Należy napisać program sterujący robotem, który pozwoli na przejście labiryntu i odtworzenie ścieżki przejścia (tam i z powrotem) na podstawie zapamiętanych na stosie danych.

Oczywiście, algorytm znajduje drogę przechodząc przez niemal wszystkie korytarze, wycofując się ze ślepych części i zdejmując ze stosu nieprawidłowe kierunki.

Stos dla programu składa się z rekordów typu **`element`** tworzonych dynamicznie procedurą **`push(x)`**, składających się
z wartości kąta skrętu **`x`** oraz wskaźnika do poprzedniego elementu.
Funkcja usuwająca/odczytująca nazywa się **`pop()`**.
Wierzchołek stosu jest wskazywany wskaźnikiem o nazwie **`top`**.




```
TYPE
  element = record
              angle : integer;
              next  : ^element;
            end;

PROCEDURE push(x : integer)
VAR temp : ^element;
BEGIN
  temp:=top;
  new(top);
  top^.angle:=x;
  top^.next:=temp;
END

FUNCTION pop():integer
VAR temp : ^element;
BEGIN
  pop:=top^.angle;
  temp:=top^.next;
  dispose(top);
  top^.next:=temp;
END
```

W procedurze **`push()`**
i funkcji **`pop()`** wskaźnik
wierzchołka stosu **`top`**
jest dostępny jako zmienna globalna.

```
PROGRAM maze();

TYPE
  frontview = (wall, exit);
  element = record
              angle : integer;
              next  : ^element;
            end
VAR
  dir : integer;
  top : ^element;

PROCEDURE go()
PROCEDURE turn(s:integer)
FUNCTION  front():frontview

PROCEDURE push(x : integer)
FUNCTION pop():integer

BEGIN
  top:=nil;
  push(0);

  REPEAT

    go();

    turn(90);
    push(pop()+90);
    WHILE front()=wall
    BEGIN
      turn(-90); {następny kierunek)
      push(pop()+90);
    END

    dir=pop();
    IF dir<360 THEN
    BEGIN
          push(dir);
          push(0);
    END

  UNTIL front()=exit

  WHILE top <> nil DO
    writeln(360-pop());
END.
```

inicjowanie stosu: musi być pierwszy element

dojdź do skrzyżowania

znajdź pierwszy wolny kierunek (korytarz) poczynając od prawej i ustaw robota w jego kierunku

jeśli to nie kierunek powrotny, zapamiętaj go i inicjuj nowy ruch (nowy element na stosie)

sprawdź czy to wyjście

wydrukuj drogę powrotną z labiryntu